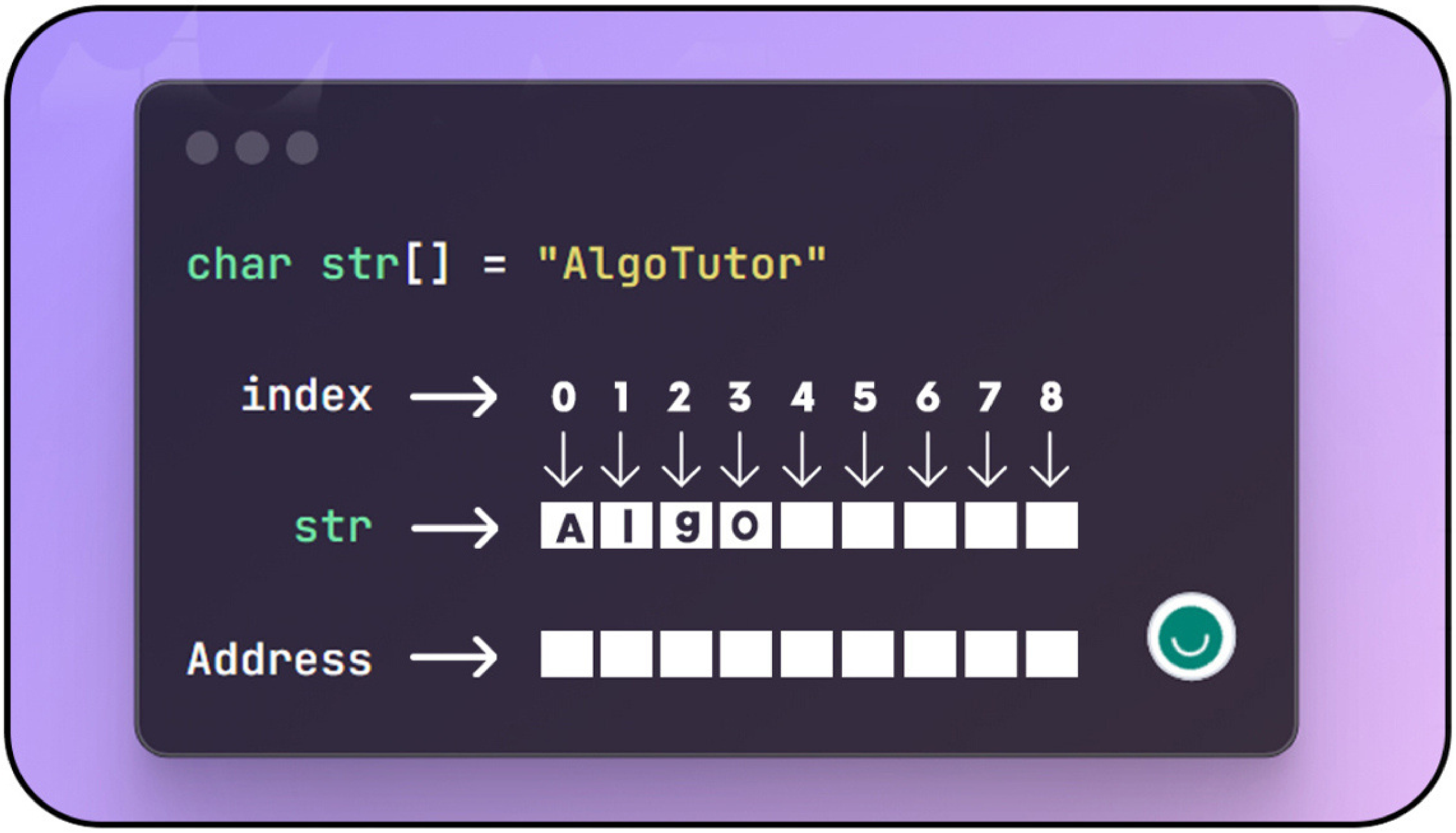
AlgoTutor

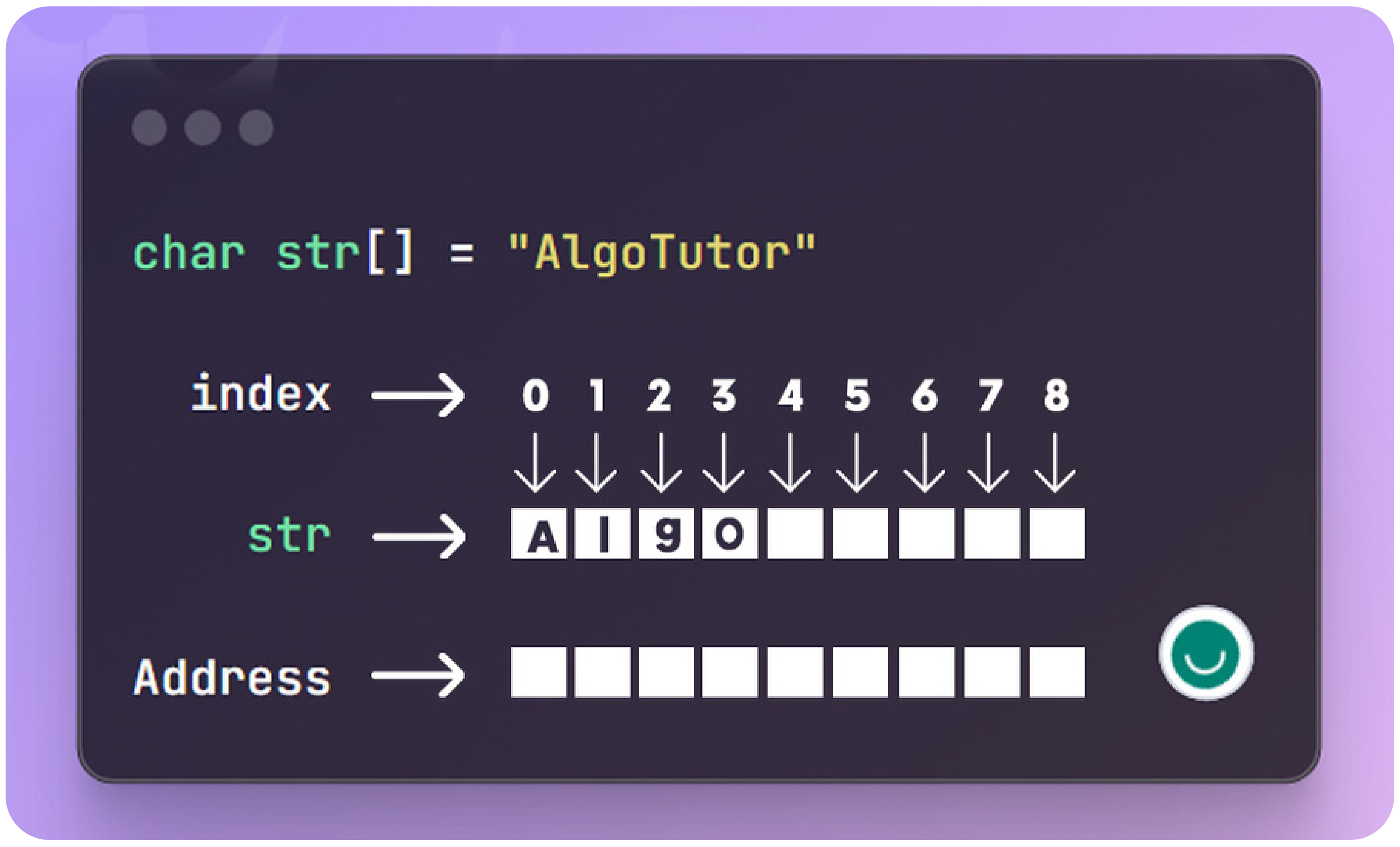
MASTER STRING IN JUST 05 DAYS


```
char str[] = "AlgoTutor"
```

Day 1

Understanding Strings

- Learn the basics of strings, including string representation and operations.
- Understand string immutability and how to manipulate strings.



```
char str[] = "AlgoTutor"
```

Day 2

String Searching and Pattern Matching

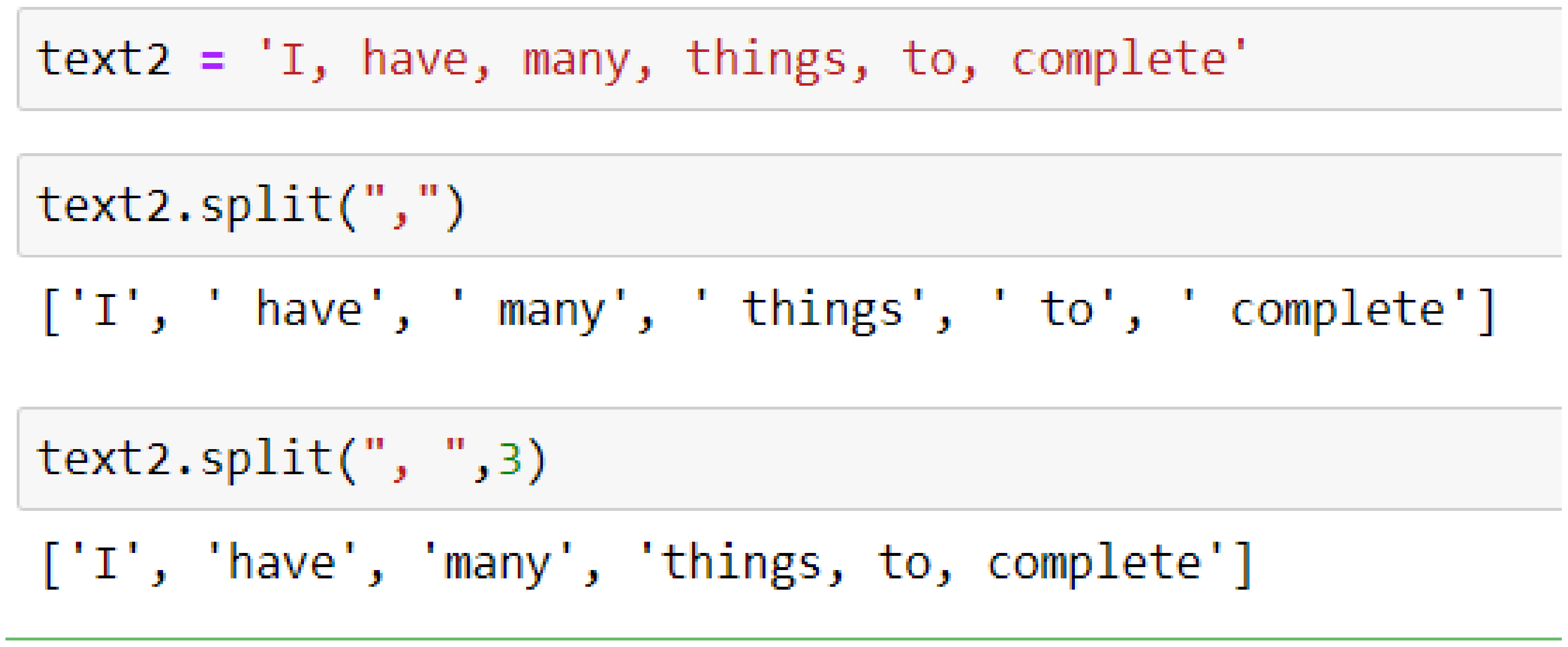
- Study string searching algorithms like brute force and the Knuth-Morris-Pratt (KMP) algorithm.
- Practice pattern matching in strings.

AlgoTutor

Day 3

String Manipulation

- Explore string manipulation techniques such as reversing a string, trimming, and formatting.
- Implement string manipulation functions.



```
text2 = 'I, have, many, things, to, complete'
```

```
text2.split(",")
```

```
['I', ' have', ' many', ' things', ' to', ' complete']
```

```
text2.split(", ",3)
```

```
['I', 'have', 'many', 'things, to, complete']
```

Day 4

Regular Expressions

- Learn about regular expressions and their uses in string processing.
- Practice writing and using regular expressions for pattern matching.

AlgoTutor

Day 5

Advanced String Problems

- Challenge yourself with advanced string-related problems.
- Apply your knowledge to real-world scenarios.

AlgoTutor

Important Practice Questions

01. Reverse Words in a String

Given an input string s, reverse the order of the words.
A word is defined as a sequence of non-space characters.
The words in s will be separated by at least one space.

Return a string of the words in reverse order concatenated by a single space.

Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:
Input: s = "the sky is blue"
Output: "blue is sky the"

Practice

02. Longest Palindromic Substring

Given a string s, return the longest palindromic substring in s.

Example 1:

Input: s = "babad"

Output: "bab"

Explanation: "aba" is also a valid answer.

Example 2:

Input: s = "cbbd"

Output: "bb"

Practice

03. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:
Input: strs = ["flower","flow","flight"]
Output: "fl"

Example 2:
Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.

Practice

AlgoTutor

04. Repeated String Match

Given two strings a and b, return the minimum number of times you should repeat string a so that string b is a substring of it. If it is impossible for b to be a substring of a after repeating it, return -1.

Notice: string "abc" repeated 0 times is "", repeated 1 time is "abc" and repeated 2 times is "abcabc".

Example 1:
Input: a = "abcd", b = "cdabcdab"
Output: 3
Explanation: We return 3 because by repeating a three times "abcdabcdabcd", b is a substring of it.

Example 2:
Input: a = "a", b = "aa"
Output: 2

Practice

05. Valid Anagram

Given two strings s and t, return true if t is an anagram of s, and false otherwise.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:
Input: s = "anagram", t = "nagaram"
Output: true

Example 2:
Input: s = "rat", t = "car"
Output: false

Practice

06. Valid Palindrome II

Given a string s, return true if the s can be palindrome after deleting at most one character from it.

Example 1:

Input: s = "aba"

Output: true

Practice

07. Integer to English Words

Convert a non-negative integer num to its English words representation.

Example 1:

Input: num = 123

Output: "One Hundred Twenty Three"

Practice

08. Group Anagrams

Given an array of strings strs, group the anagrams together. You can return the answer in any order.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

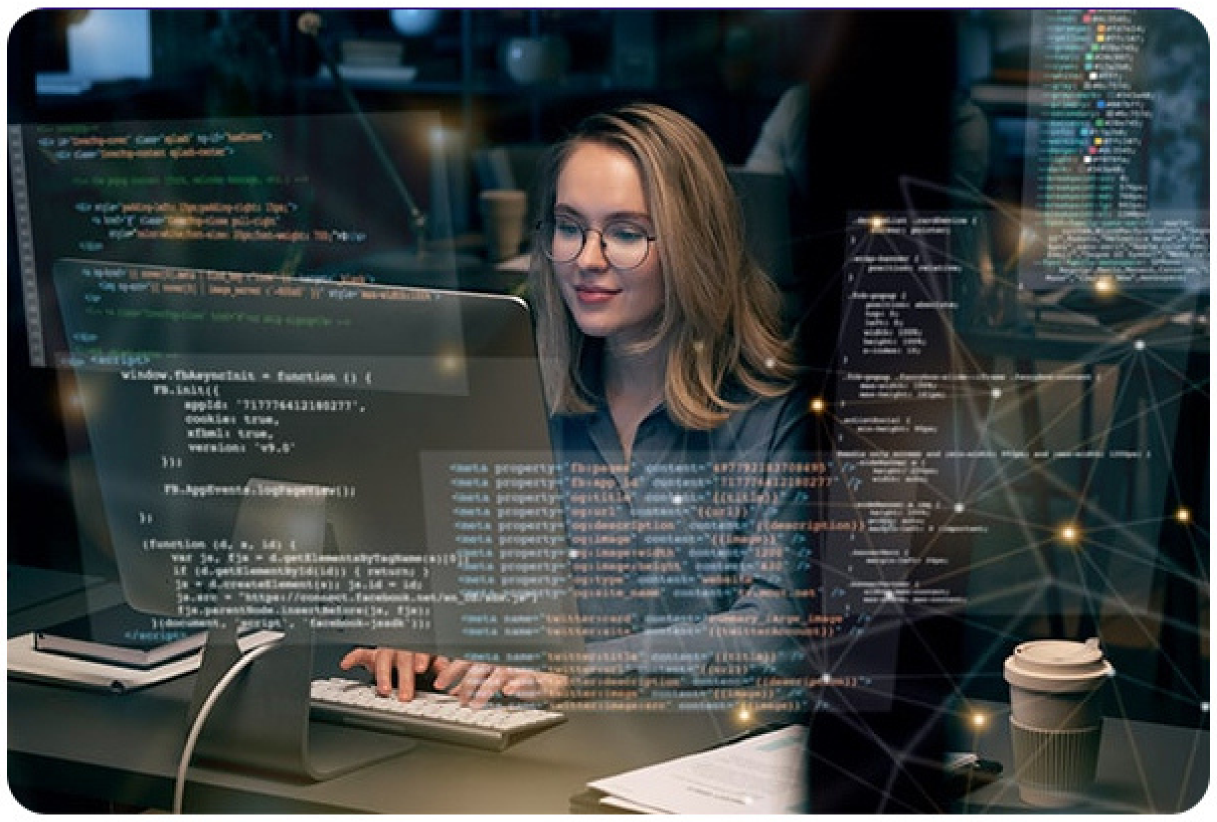
Example 2:
Input: strs = [""]
Output: [[""]]

Example 3:
Input: strs = ["a"]
Output: [["a"]]

Practice

AlgoTutor

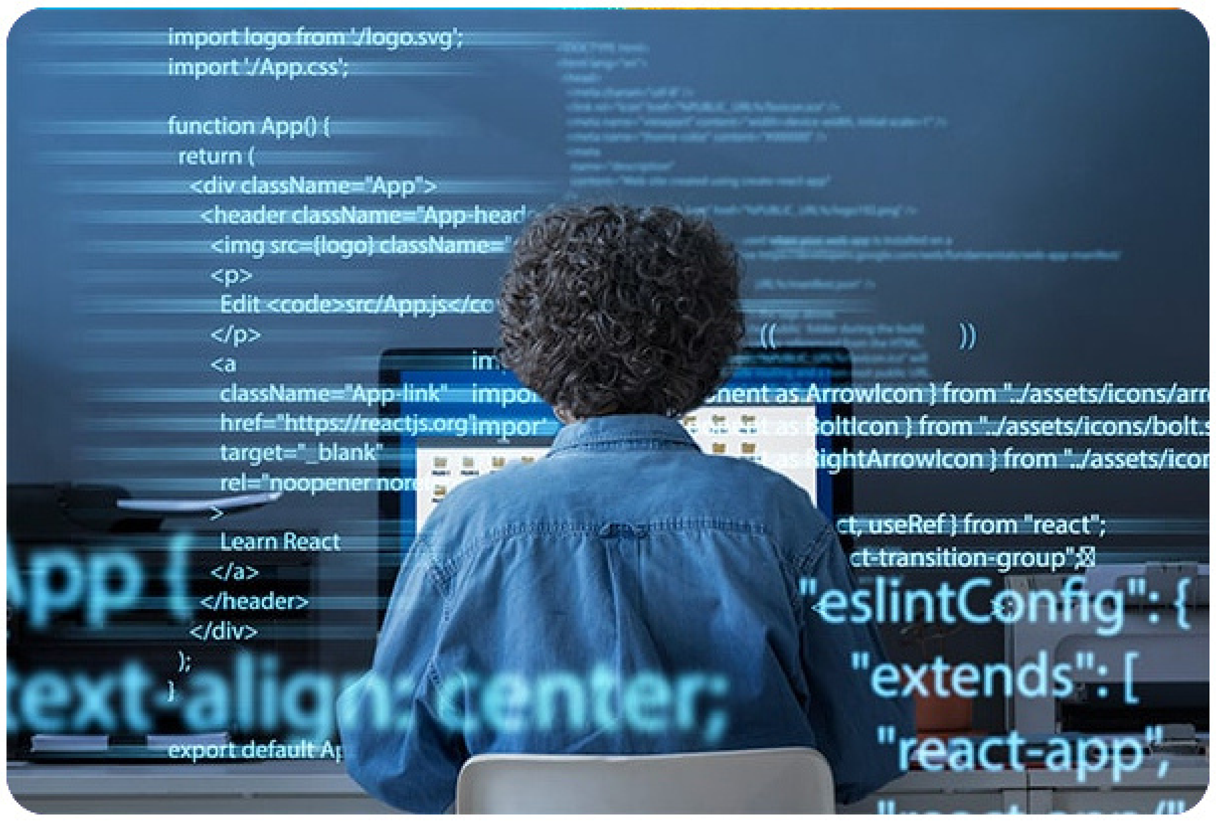
Take The First Step Towards Fulfilling a Career



Mastering DSA & System Design

Advance Data Science & ML

Full Stack Web Development - MERN



Mastering DSA & System Design with Full Stack Specialization

For Admission Enquiry +91 - 72600 58093

AlgoTutor

09. Basic Calculator II

Given a string s which represents an expression, evaluate this expression and return its value.

The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of [-231, 231 - 1].

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

Example 1:
Input: s = "3+2*2"
Output: 7

Example 2:
Input: s = " 3/2 "
Output: 1

Practice

AlgoTutor

10. Text Justification

Given an array of strings words and a width maxWidth, format the text such that each line has exactly maxWidth characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces ' ' when necessary so that each line has exactly maxWidth characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left-justified, and no extra space is inserted between words.

Practice

11. Remove Outermost Parentheses

A valid parentheses string is either empty "", "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation.

- For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings.

A valid parentheses string s is primitive if it is nonempty, and there does not exist a way to split it into s = A + B, with A and B nonempty valid parentheses strings.
Given a valid parentheses string s, consider its primitive decomposition: s = P1 + P2 + ... + Pk, where Pi are primitive valid parentheses strings.
Return s after removing the outermost parentheses of every primitive string in the primitive decomposition of s.

Example 1:
Input: s = "(()())(())"
Output: "()()()"

Practice

12. Largest Odd Number in String

You are given a string num, representing a large integer. Return the largest-valued odd integer (as a string) that is a non-empty substring of num, or an empty string "" if no odd integer exists.

A substring is a contiguous sequence of characters within a string.

Example 1:
Input: num = "52"
Output: "5"
Explanation: The only non-empty substrings are "5", "2", and "52". "5" is the only odd number.

Example 2:
Input: num = "4206"
Output: ""
Explanation: There are no odd numbers in "4206".

Practice

13. Isomorphic Strings

Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

Example 1:
Input: s = "egg", t = "add"
Output: true

Example 2:
Input: s = "foo", t = "bar"
Output: false

Practice

14. Rotate String

Given two strings s and goal, return true if and only if s can become goal after some number of shifts on s.

A shift on s consists of moving the leftmost character of s to the rightmost position.

- For example, if s = "abcde", then it will be "bcdea" after one shift.

Example 1:
Input: s = "abcde", goal = "cdeab"
Output: true

Example 2:
Input: s = "abcde", goal = "abced"
Output: false

Practice

15. Sort Characters By Frequency

Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them.

Example 1:
Input: s = "tree"
Output: "eert"

Example 2:
Input: s = "cccaaa"
Output: "aaaccc"

Example 3:
Input: s = "Aabb"
Output: "bbAa"

Practice

16. Maximum Nesting Depth of the Parentheses

A string is a valid parentheses string (denoted VPS) if it meets one of the following:

- It is an empty string "", or a single character not equal to "(" or ")",
- It can be written as AB (A concatenated with B), where A and B are VPS's, or
- It can be written as (A), where A is a VPS.

We can similarly define the nesting depth depth(S) of any VPS S as follows:

- depth("") = 0
- depth(C) = 0, where C is a string with a single character not equal to "(" or ")".
- depth(A + B) = max(depth(A), depth(B)), where A and B are VPS's.
- depth("(" + A + ")") = 1 + depth(A), where A is a VPS.

Example 1:

Input: s = "(1+(2*3)+((8)/4))+1"

Output: 3

Practice

17. Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Example 1:

Input: s = "III"

Output: 3

Example 2:

Input: s = "LVIII"

Output: 58

Practice

AlgoTutor

18. String to Integer (atoi)

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for myAtoi(string s) is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).

Example 1:

Input: s = "42"

Output: 42

Practice

AlgoTutor

19. Sum of Beauty of All Substrings

The beauty of a string is the difference in frequencies between the most frequent and least frequent characters.

- For example, the beauty of "abaacc" is 3 - 1 = 2.

Given a string s, return the sum of beauty of all of its substrings.

Example 1:

Input: s = "aabcb"

Output: 5

Example 2:

Input: s = "aabcbaa"

Output: 17

Practice

20. Word Break II

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in any order.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:
Input: s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]
Output: ["cats and dog","cat sand dog"]

Example 2:
Input: s = "pineapplepenapple", wordDict = ["apple","pen","applepen","pine","pineapple"]
Output: ["pine apple pen apple","pineapple pen apple","pine applepen apple"]

Practice

AlgoTutor

WHY ALGOTUTOR

- 100% Placement Assistance
- 1-1 personal mentorship from Indusrty experts
- 200+ Successful Alumni
- 147(Avg.)% Salary Hike
- 100% Success Rate
- 23 LPA (Avg.) CTC
- Learn from scratch
- Career Services

For Admission Enquiry

+91-7260058093

info@algotutor.io